Dialogue & Discourse 7(3) (2016) 65–88 doi: 10.5087/dad.2016.303

Recurrent Polynomial Network for Dialogue State Tracking*

Kai Sun KS985@CORNELL.EDU
Department of Computer Science
Cornell University, Ithaca, NY 14853, USA

Qizhe Xie CHEEZER@SJTU.EDU.CN
Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering
SpeechLab, Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, 200240, China

Kai Yu† KAI.YU@SJTU.EDU.CN
Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering
SpeechLab, Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, 200240, China

Editor: Jason D. Williams, Antoine Raux, and Matthew Henderson

Submitted 04/15; Accepted 02/16; Published online 04/16

Abstract

Dialogue state tracking (DST) is a process to estimate the distribution of the dialogue states as a dialogue progresses. Recent studies on constrained Markov Bayesian polynomial (CMBP) framework take the first step towards bridging the gap between rule-based and data-driven approaches for DST. In this paper, a novel hybrid framework – *recurrent polynomial network* (RPN) is proposed to further improve the combination of rule-based and data-driven approaches. RPN's unique structure enables the framework to have all the advantages of CMBP including efficiency, portability and interpretability. Additionally, RPN achieves more properties of data-driven approaches than CMBP. RPN was evaluated on the data corpora of the second and the third Dialog State Tracking Challenge (DSTC-2/3). Experiments showed that RPN can significantly outperform both traditional rule-based approaches and data-driven statistical approaches with similar feature set. Compared with the state-of-the-art data-driven statistical DST approaches with a lot richer features, RPN is also competitive.

Keywords: Statistical Dialogue Management, Dialogue State Tracking, Recurrent Polynomial Network

1. Introduction

A task-oriented spoken dialogue system (SDS) is a system that can interact with a user to accomplish a predefined task through speech. It usually has three modules: input, output and control, shown in Figure 1. The input module consists of automatic speech recognition (ASR) and spoken language

*. This work was supported by the Program for Professor of Special Appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning and the China NSFC project No. 61222208.

†. Corresponding author

understanding (SLU), with which the user speech is converted into text and semantics-level user dialogue acts are extracted. Once the user dialogue acts are received, the control module, also called dialogue management accomplishes two missions. One mission is called dialogue state tracking (DST), which is a process to estimate the distribution of the dialogue states, an encoding of the machine's understanding about the conversion as a dialogue progresses. Another mission is to choose semantics-level machine dialogue acts to direct the dialogue given the information of the dialogue state, referred to as dialogue decision making. The output module converts the machine acts into text via natural language generation and generates speech according to the text via text-to-speech synthesis.

Dialogue management is the core of a SDS. Traditionally, dialogue states are assumed to be observable and hand-crafted rules are employed for dialogue management in most commercial SDSs. However, because of unpredictable user behaviour, inevitable ASR and SLU errors, dialogue state tracking and decision making are difficult (Williams and Young, 2007). Consequently, in recent years, there is a research trend from rule-based dialogue management towards statistical dialogue management. Partially observable Markov decision process (POMDP) framework offers a well-founded theory to both dialogue state tracking and decision making in statistical dialogue management (Roy et al., 2000; Zhang et al., 2001; Williams and Young, 2005, 2007; Thomson and Young, 2010; Gašić and Young, 2011; Young et al., 2010). In previous studies of POMDP, dialogue state tracking and decision making are usually investigated together. In recent years, to advance the research of statistical dialogue management, the DST problem is raised out of the statistical dialogue management framework so that a bunch of models can be investigated for DST.

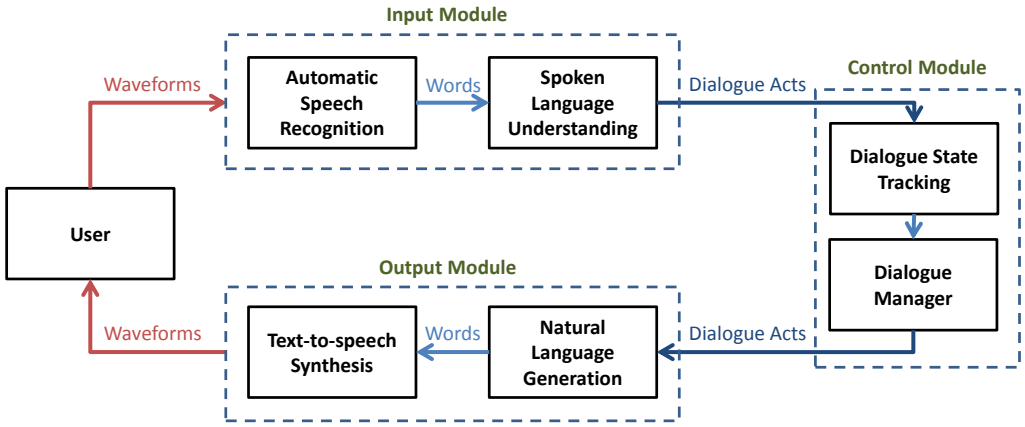

Figure 1: Diagram of a spoken dialogue system (SDS)

Most early studies of POMDP on DST were devoted to generative models (Young et al., 2010), which learn joint probability distribution over observation and labels. Fundamental weaknesses of generative model was revealed by the result of (Williams, 2012). In contrast, discriminative state tracking models have been successfully used for SDSs (Deng et al., 2013). Compared to generative models where assumptions about probabilistic dependencies of features are usually needed, discriminative models directly model probability distribution of labels given observation, enabling rich features to be incorporated. The results of the Dialog State Tracking Challenge (DSTC) (Williams et al., 2013; Henderson et al., 2014b,a) further demonstrated the power of discriminative statisti-

cal models, such as Maximum Entropy (MaxEnt) (Lee and Eskenazi, 2013), Conditional Random Field (Lee, 2013), Deep Neural Network (DNN) (Sun et al., 2014a), and Recurrent Neural Network (RNN) (Henderson et al., 2014d). In addition to discriminative statistical models, discriminative rule-based models have also been investigated for DST due to their efficiency, portability and interpretability and some of them showed good performance and generalisation ability in DSTC (Zilka et al., 2013; Wang and Lemon, 2013). However, both rule-based and statistical approaches have some disadvantages. Statistical approaches have shown large variation in performance and poor generalisation ability due to the lack of data (Williams, 2012). Moreover, statistical models usually have more complex model structure and features than rule-based models, and thus can hardly achieve efficiency, portability and interpretability as rule-based models. As for rule-based models, their performance is usually not competitive to the best data-driven statistical approaches. Additionally, since they require lots of expert knowledge and there is no general way to design rule-based models with prior knowledge, they are typically difficult to design and maintain. Furthermore, there lacks a way to improve their performance when training data are available.

Recent studies on constrained Markov Bayesian polynomial (CMBP) framework take the first step towards bridging the gap between rule-based and data-driven approaches for DST (Sun et al., 2014b; Yu et al., 2015). CMBP formulate rule-based DST in a general way and allow data-driven rules to be generated. Concretely, in the CMBP framework, DST models are defined as polynomial functions of a set of features whose coefficients are integer and satisfy a set of constraints where prior knowledge is encoded. The optimal DST model is selected by evaluating each model on training data. Yu et al. (2015) further extended CMBP to real-coefficient polynomial where the real coefficients can be estimated by optimizing the DST performance on training data using grid search. CMBP offers a way to improve the performance when training data are available and achieves competitive performance to the state-of-the-art data-driven statistical approaches, while at the same time keeping most of the advantages of rule-based models. Nevertheless, adding features to CMBP is not as easy as to most data-driven statistical approaches because on the one hand, the features usually need to be probability related features, on the other hand, additional prior knowledge is needed to constrain the search space. For the same reason, increasing the model complexity, such as by using higher-order polynomial, by introducing hidden variables, etc. also requires additional suitable prior knowledge to be introduced to limit the search space. Moreover, CMBP can hardly fully utilize the labelled data because in practice its polynomial coefficients are set by grid search.

In this paper, a novel hybrid framework, referred to as *recurrent polynomial network* (RPN), is proposed to further improve the combination of rule-based and data-driven approaches for DST. Although the basic idea for transforming rules to neural networks has been there for many years (Cloete and Zurada, 2000), few work has been done for dialogue state tracking. RPN can be regarded as a kind of human interpretable computation network, and its unique structure enables the framework to have all the advantages of CMBP including efficiency, portability and interpretability. Additionally, RPN achieves more properties of data-driven approaches than CMBP. In general, RPN has neither restriction to feature type, nor search space issue to be concerned about, so adding features and increasing the model complexity are much easier in RPN. Furthermore, RPN can better explore the parameter space than CMBP with labelled data.

The DSTCs have provided the first common testbed in a standard format, along with a suite of evaluation metrics to facilitate direct comparisons among DST models (Williams et al., 2013). To evaluate the effectiveness of RPN for DST, both the dataset from the second Dialog State Tracking Challenge (DSTC-2) which is in restaurants domain (Henderson et al., 2014b) and the dataset from

the third Dialog State Tracking Challenge (DSTC-3) which is in tourists domain (Henderson et al., 2014a) are used. For both of the datasets, the dialogue state tracker receives SLU N-best hypotheses for each user turn, each hypothesis having a set of act-slot-value tuples with a confidence score. The dialogue state tracker is supposed to output a set of distributions of the dialogue state. In this paper, only joint goal tracking, which is the most difficult and general task of DSTC-2/3, is of interest.

The rest of the paper is organized as follows. Section 2 discusses ways of combining rule-based and data-driven approaches. Section 3 formulates RPN. The RPN framework for DST is described in section 4, followed by experiments in section 5. Finally, section 6 concludes the paper.

2. Combining Rule-based and Data-driven Approaches

Broadly, it is straightforward to come up with two possible ways to combine rule-based and data-driven approaches: one starts from rule-based models, while the other starts from data-driven models. CMBP takes the first way, which is derived as an extension of a rule-based model (Sun et al., 2014b; Yu et al., 2015). Inspired by the observation that many rule-based models such as models proposed by Wang and Lemon (2013) and Zilka et al. (2013) are based on Bayes' theorem, in the CMBP framework, a DST rule is defined as a polynomial function of a set of probabilities since Bayes' theorem is essentially summation and multiplication of probabilities. Here, the polynomial coefficients can be seen as parameters. To make the model have good DST performance, prior knowledge or intuition is encoded to the polynomial functions by setting certain constraints to the polynomial coefficients, and the coefficients can further be optimized by data-driven optimization. Therefore, starting from rule-based models, CMBP can directly incorporate prior knowledge or intuition into DST, while at the same time, the model is allowed to be data-driven.

More concretely, assuming that both slot and value are independent, a CMBP model can be defined as

$$\begin{aligned} b_{t+1}(v) = \quad & \mathcal{P}\left(b_t(v), P^+_{t+1}(v), P^-_{t+1}(v), \tilde{P}^+_{t+1}(v), \tilde{P}^-_{t+1}(v), 1, b^r_t\right) \\ & \texttt{s.t. constraints} \end{aligned} \tag{1}$$

where $b_t(v)$, $P^+_t(v)$, $P^-_t(v)$, $\tilde{P}^+_t(v)$, $\tilde{P}^-_t(v)$, b^r_t are all probabilistic *features* which are defined as below:

- $b_t(v)$: belief of "the value being v at turn t"
- $P^+_t(v)$: sum of scores of SLU hypotheses informing or affirming value v at turn t
- $P^-_t(v)$: sum of scores of SLU hypotheses denying or negating value v at turn t
- $\tilde{P}^+_t(v) = \sum_{v' \notin \{v, \texttt{None}\}} P^+_t(v')$
- $\tilde{P}^-_t(v) = \sum_{v' \notin \{v, \texttt{None}\}} P^-_t(v')$
- b^r_t: belief of the value being 'None' (the value not mentioned) at turn t, i.e. $b^r_t = 1 - \sum_{v' \neq \texttt{None}} b_t(v')$

and $\mathcal{P}(\cdot)$ is a multivariate polynomial function[1]

$$\mathcal{P}(\iota_0, \cdots, \iota_D) = \sum_{0 \le k_1 \le \cdots \le k_n \le D} g_{k_1, \cdots, k_n} \prod_{1 \le i \le n} \iota_{k_i} \tag{2}$$

where $D+1$ is the number of input variables, n is the order of the polynomial, $g_{k_1,\cdots,k_n}$ is the *parameter* of CMBP. Order 3 gives good trade-off between complexity and performance, hence order 3 is used in our previous work (Sun et al., 2014b; Yu et al., 2015) and this paper. Since both slot independence and value independence are assumed in this paper, the belief of the joint goal $(slot_1 = v_1, slot_2 = v_2, \ldots, slot_N = v_N)$ is $\prod_{i=1}^{N} b(v_i)$.

The *constraints* in equation (1) encode all necessary probabilistic conditions (Yu et al., 2015). For instance,

$$0 \le P_t^+(v) + \tilde{P}_t^+(v) \le 1 \tag{3}$$

$$0 \le b_t(v) \le 1 \tag{4}$$

The *constraints* in equation (1) also encode prior knowledge or intuition (Yu et al., 2015). For example, the rule *"goal belief should be unchanged or positively correlated with the positive scores from SLU"* can be represented by

$$\frac{\partial \mathcal{P}\left(b_t(v), P_{t+1}^+(v), P_{t+1}^-(v), \tilde{P}_{t+1}^+(v), \tilde{P}_{t+1}^-(v), 1, b_t^r\right)}{\partial P_{t+1}^+(v)} \ge 0 \tag{5}$$

The definition of CMBP formulates a search space of rule-based models, where it is easy to employ data-driven criterion to find a rule-based model with good performance. Considering CMBP is originally motivated from Bayesian probability operation which leads to the natural use of integer polynomial coefficients ($g \in \mathbb{Z}$), the data-driven optimization can be formulated by an integer programming program (Sun et al., 2014b; Yu et al., 2015). Additionally, CMBP can also be viewed as a data-driven approach. Hence, the polynomial coefficients can be extended to real numbers. The optimization of real-coefficient can be done by first getting an integer-coefficient CMBP and then performing hill climbing search (Yu et al., 2015).

3. Recurrent Polynomial Network

Recurrent polynomial network, which is proposed in this paper, takes the other way to combine rule-based and data-driven approaches. The basic idea of RPN is to enable a kind of data-driven model to take advantage of prior knowledge or intuition by using the parameters of rule-based models to initialize the parameters of data-driven models.

Computational networks have been researched for decades from very basic architectures such as perceptron (Rosenblatt, 1958) to today's various kinds of deep neural networks. Recurrent computational networks, a class of computational networks which have recurrent connections, have also been researched for a long time, from fully recurrent networks to networks with relatively complex structures such as long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997). Like common neural networks, RPN is a data-driven approach so it is as easy to add features and try

1. The notation of $\sum_{0 \le k_1 \le \cdots \le k_n \le D}$ is shorthand for series of n nested sums over ranges bounded, i.e. $\sum_{0 \le k_1 \le D} \sum_{k_1 \le k_2 \le D} \cdots \sum_{k_{n-1} \le k_n \le D}$.

complex structures in RPN as in neural networks. However, compared with common neural networks which are "black boxes", an RPN can essentially be seen as a polynomial function. Hence, considering that a CMBP is also a polynomial function, the encoded prior knowledge and intuition in CMBP can be transferred to RPN by using the parameters of CMBP to initialize RPN. In this way, it combines rule-based models and data-driven models.

A recurrent polynomial network is a computational network. The network contains multiple edges and loops. Each node is either an *input node*, which is used to represent an input value, or a *computation node*. Each node x is set an initial value $u_x^{(0)}$ at time 0, and its value is updated at time $1, 2, \cdots$. Both the type of edges and the type of nodes decide how the nodes' values are updated. There are two types of edges. One type, referred to as *type-1*, indicates the value updating at time t takes the value of a node at time $t-1$, i.e. type-1 edges are recurrent edges, while the other type, referred to as *type-2*, indicates the value updating at time t takes another node's value at time t. Except for loops made of "type-1" edges, the network should not have loops. For simplicity, let I_x be the set of nodes y which are linked to node x by a type-1 edge, $\hat{I}_x$ be the set of nodes y which are linked to node x by a type-2 edge. Based on these definitions, two types of computation nodes, *sum* and *product*, are introduced. Specifically, at time $t > 0$, if node x is a sum node, its value $u_x^{(t)}$ is updated by

$$u_x^{(t)} = \sum_{y \in I_x} w_{x,y} u_y^{(t-1)} + \sum_{y \in \hat{I}_x} \hat{w}_{x,y} u_y^{(t)} \tag{6}$$

where $w, \hat{w} \in \mathbb{R}$ are the weights of edges.

Similarly, if node x is a product node, its value is updated by

$$u_x^{(t)} = \prod_{y \in I_x} {u_y^{(t-1)}}^{M_{x,y}} \prod_{y \in \hat{I}_x} {u_y^{(t)}}^{\hat{M}_{x,y}} \tag{7}$$

where $M_{x,y}$ and $\hat{M}_{x,y}$ are integers, denoting the multiplicity of the type-1 edge $\overrightarrow{yx}$, and the multiplicity of the type-2 edge $\overrightarrow{yx}$ respectively. It is noted that only $w, \hat{w}$ are parameters of RPN while $M_{x,y}, \hat{M}_{x,y}$ are constant given the structure of an RPN.

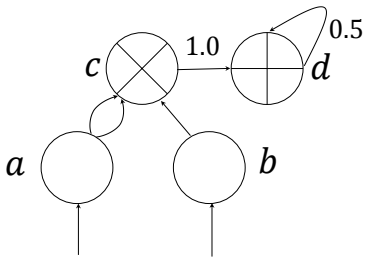

Figure 2: A simple example of RPN. The type of nodes a, b, c, d are input, input, product, and sum respectively. Edge $\overrightarrow{dd}$ is of type-1, while the other edges are of type-2. $\hat{M}_{a,c} = 2$, $\hat{M}_{b,c} = \hat{M}_{c,d} = M_{d,d} = 1$.

Let $\boldsymbol{u}^{(t)}$, $\hat{\boldsymbol{u}}^{(t)}$ denote the vector of computation nodes' values and the vector of input nodes' values at time t respectively, then a well-defined RPN can be seen as a polynomial function as

below.

$$\boldsymbol{u}^{(t)} = \mathcal{P}\left(\hat{\boldsymbol{u}}^{(t)}, \boldsymbol{u}^{(t-1)}, 1\right) \tag{8}$$

where $\mathcal{P}$ is defined by equation (2). For example, for the RPN in figure 2, its corresponding polynomial function is

$$\begin{aligned}(u_c^{(t)}, u_d^{(t)}) =& \mathcal{P}\left(u_a^{(t)}, u_b^{(t)}, u_c^{(t-1)}, u_d^{(t-1)}, 1\right) \\ =& \left((u_a^{(t)})^2 u_b^{(t)}, 0.5u_d^{(t-1)} + (u_a^{(t)})^2 u_b^{(t)}\right)\end{aligned} \tag{9}$$

Each computation node can be regarded as an *output node*. For example, for the RPN in figure 2, node c and node d can be set as output nodes.

4. RPN for Dialogue State Tracking

As introduced in section 1, in this paper, the dialogue state tracker receives SLU N-best hypotheses for each user turn, each hypothesis having a set of act-slot-value tuples with a confidence score. The dialogue state tracker is supposed to output a set of distributions over the joint user goal, i.e., the value for each slot. For simplicity and consistency with the work of Sun et al. (2014b) and Yu et al. (2015), slot and value independence are assumed in the RPN model for dialogue state tracking[2], though neither CMBP nor RPN is limited to the assumptions. In the rest of the paper, $b_t(v), P_t^+(v), P_t^-(v), \tilde{P}_t^+(v), \tilde{P}_t^-(v)$ are abbreviated by $b_t, P_t^+, P_t^-, \tilde{P}_t^+, \tilde{P}_t^-$ respectively in circumstances where there is no ambiguity.

4.1 Structure

Before describing details of the structure used in the real situations, to help understand the corresponding relationship between RPN and CMBP, let's first look at a simplified case with a smaller feature set and a smaller order, which is a corresponding relationship between the RPN shown in figure 3 and 2-order polynomial (10) with features $b_{t-1}, P_t^+, 1$:

$$\begin{aligned}b_t &= 1 - (1 - b_{t-1})(1 - P_t^+) \\ &= b_{t-1} + P_t^+ - P_t^+ b_{t-1}\end{aligned} \tag{10}$$

Recall that a CMBP of polynomial order 2 with 3 features is the following equation (refer to equation (2)):

$$\mathcal{P}(\iota_0, \iota_1, \iota_2) = \sum_{0 \le k_1 \le k_2 \le 2} g_{k_1,k_2} \prod_{1 \le i \le 2} \iota_{k_i} \tag{11}$$

2. For DSTC-2/3 tasks, one slot can have at most one value, i.e. $0 \le \sum_{v \neq \texttt{None}} b_t(v) \le 1$. Since value independence is assumed, to strictly maintain that relation, the belief is rescaled to ensure the sum of the belief of each value plus the belief of 'None' to be 1 when the belief is being output. Actually, to enable RPN strictly maintain that relation, our original design of RPN had a "normalization" step when passing the belief from turn t to turn $t+1$. The "normalization" step will rescale the belief to make the sum of the belief of each value plus the belief of 'None' to be 1. Our later experiment, however, demonstrated that there was no significant performance difference between the RPN with and without the "normalization" step. Therefore, in practice, for simplicity, the "normalization" step can be omitted, value independence can be assumed, and the only thing needed is to rescale the belief when it is being output.

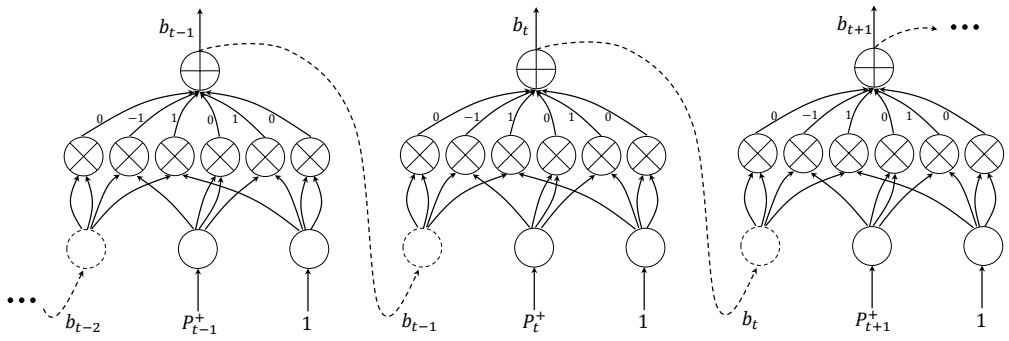

Figure 3: A simple example of RPN for DST.

The RPN in figure 3 has three layers. The first layer only contains input nodes. The second layer only contains product nodes. The third layer only contains sum nodes. Every product node in the second layer denotes a monomial of order 2 such as $(b_{t-1})^2, b_{t-1}, P_t^+$ and so on. Every product node in the second layer is linked to the sum node in the third layer whose value is a weighted sum of value of product nodes. With weight set according to coefficients in equation (10), the value of sum node in the third layer is essentially the b_t in equation (10).

Like the above simplified case, a layered RPN structure shown in figure 4 is used for dialogue state tracking in our first trial which essentially corresponds to 3-order CMBP, though the RPN framework is not limited to the layered topology. Recall that a CMBP of polynomial order 3 is used as shown in the following equation (refer to equation (2)):

$$\mathcal{P}(\iota_0, \cdots, \iota_D) = \sum_{0 \le k_1 \le k_2 \le k_3 \le D} g_{k_1,k_2,k_3} \prod_{1 \le i \le 3} \iota_{k_i} \tag{12}$$

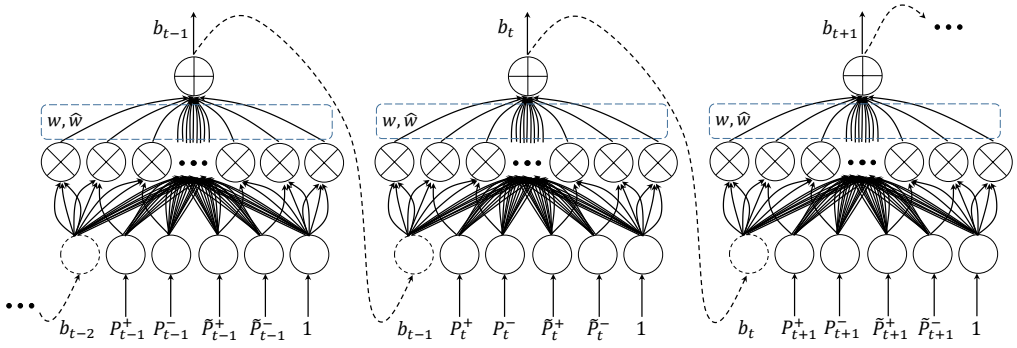

Figure 4: RPN for DST.

Let (l, i) denote the index of i-th node in the l-th layer. The detailed definitions of each layer are as follows:

- First layer / Input layer:

  Input nodes are features at turn t, which corresponds to variables in CMBP in section 2. i.e.

- $u^{(t)}_{(1,0)} = b_{t-1}$
- $u^{(t)}_{(1,1)} = P^+_t$
- $u^{(t)}_{(1,2)} = P^-_t$
- $u^{(t)}_{(1,3)} = \tilde{P}^+_t$
- $u^{(t)}_{(1,4)} = \tilde{P}^-_t$
- $u^{(t)}_{(1,5)} = 1$

While 7 features are used in previous work of CMBP (Sun et al., 2014b; Yu et al., 2015), only 6 of them are used in RPN with feature b^r_{t-1} removed[3]. Since our experiments showed the performance of CMBP would not become worse without feature b^r_{t-1}, to make the structure more compact, b^r_{t-1} is not used in this paper for RPN. In accordance to this, CMBP mentioned in the rest of paper does not use this feature either.

- Second layer:

  The value of every product node in the second layer is a monomial like the simplified case. And every product node has indegree 3 which is corresponding to the order of CMBP.

  Every monomial in CMBP is the product of three repeatable features. Correspondingly, the value of every product node in second layer is the product of values of three repeatable nodes in the first layer. Every triple $(k_1, k_2, k_3)(0 \leq k_1 \leq k_2 \leq k_3 \leq 5)$ is enumerated to create a product node $x = (2, i)$ in second layer that nodes $(1, k_1), (1, k_2), (1, k_3)$ are linked to. i.e. $\hat{I}_x = \{(1, k_1), (1, k_2), (1, k_3)\}$. And thus $u^{(t)}_x = u^{(t)}_{1,k_1} u^{(t)}_{1,k_2} u^{(t)}_{1,k_3}$.

  And different node in the second layer is created by a distinct triple. So given the 6 input features, there are $\sum_{k_1=0}^{5} \sum_{k_2=k_1}^{5} \sum_{k_3=k_2}^{5} 1 = \binom{6+3-1}{3} = 56$ nodes in the second layer.

  To simplify the notation, a bijection from nodes to monomials is defined as:

$$\mathcal{F} : \{x | x \text{ is the index of a node in the } 2^{nd} \text{ layer}\} \rightarrow \{(k_1, k_2, k_3) | 0 \leq k_1 \leq k_2 \leq k_3 \leq D\} \tag{13}$$

$$\mathcal{F}(x) = (k_1, k_2, k_3) \Longleftrightarrow u^{(t)}_{2,i} = u^{(t)}_{1,k_1} u^{(t)}_{1,k_2} u^{(t)}_{1,k_3} \tag{14}$$

  where $D + 1 = 6$ is the number of nodes in the first layer, i.e. input feature dimension.

- Third layer:

  The value of sum node $x = (3, 0)$ in the third layer is corresponding to the output value of CMBP.

  Every product nodes in the second layer are linked to it. Node x's value $u^{(t)}_{3,0}$ is a weighted sum of values of product node $u^{(t)}_{2,i}$ where the weights correspond to g_{k_1,k_2,k_3} in equation (12).

3. b^r_t is the belief of value being 'None', whose precise definition is given in section 2.

With only sum and product operation involved, every node's value is essentially a polynomial of input features. And just like recurrent neural network, node at time t can be linked to node at time $t+1$. That is why this model is called recurrent polynomial network.

The parameters of the RPN can be set according to CMBP coefficients g_{k_1,k_2,k_3} in equation (12) so that the output value is the same as the value of CMBP, which is a direct way of applying prior knowledge and intuition to data-driven models. It is explained in detail in section 4.4.

4.2 Activation Function

In DST, the output value is a belief which should lie in $[0,1]$, while values of computational nodes are not bound by certain interval in RPN. Experiments showed that if weights are not properly set in RPN and a belief b_{t-1} output by RPN is larger than 1, then b_t may grow much larger because b_t is the weighted sum of monomials such as $(b_{t-1})^3$. Belief of later turns such as b_{t+10} will tend to infinity.

Therefore, an activation function is needed to map b_t to a legal belief value (referred to as b'_t) in $(0,1)$. 3 kinds of functions, the *logistic* function, the *clip* function, and the *softclip* function have been considered. A logistic function is defined as

$$logistic(x) = \frac{L}{1+e^{-\eta(x-x_0)}} \tag{15}$$

It can map $\mathbb{R}$ to $(0,1)$ by setting $L=1$. However, since basically the RPN designed for dialogue state tracking does similar operation as CMBP which is motivated from Bayesian probability operation (Yu et al., 2015), intuitively we expect the activation function to be linear on $[0,1]$ so that little distortion is added to the belief.

As an alternation, a *clip* function is defined as

$$clip(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } 0 \le x \le 1 \\ 1 & \text{if } x > 1 \end{cases} \tag{16}$$

It is linear on $[0,1]$. However, if $b'_t = clip(b_t)$, $b_t \notin [0,1]$ and $\mathcal{L}$ is the loss function,

$$\frac{\partial \mathcal{L}}{\partial b_t} = \frac{\partial \mathcal{L}}{\partial b'_t}\frac{\partial b'_t}{\partial b_t} = \frac{\partial \mathcal{L}}{\partial b'_t} \times 0 = 0 \tag{17}$$

Thus, $\frac{\partial \mathcal{L}}{\partial b_t}$ would be 0 whatever $\frac{\partial \mathcal{L}}{\partial b'_t}$ is. This gradient vanishing phenomenon may affect the effectiveness of backpropagation training in section 4.5.

So an activation function $softclip(\cdot)$ is introduced, which is a combination of logistic function and clip function. Let ϵ denote a small value such as 0.01, δ denote the offset of sigmoid function such that $sigmoid\,(\epsilon - 0.5 + \delta) = \epsilon$. Here the *sigmoid* function refers to the special case of the logistic function defined by the formula

$$sigmoid(x) = \frac{1}{1+e^{-x}} \tag{18}$$

The softclip function is defined as

$$softclip(x) \triangleq \begin{cases} sigmoid\,(x - 0.5 + \delta) & \text{if } x \le \epsilon \\ x & \text{if } \epsilon < x < 1-\epsilon \\ sigmoid\,(x - 0.5 - \delta) & \text{if } x \ge 1-\epsilon \end{cases} \tag{19}$$

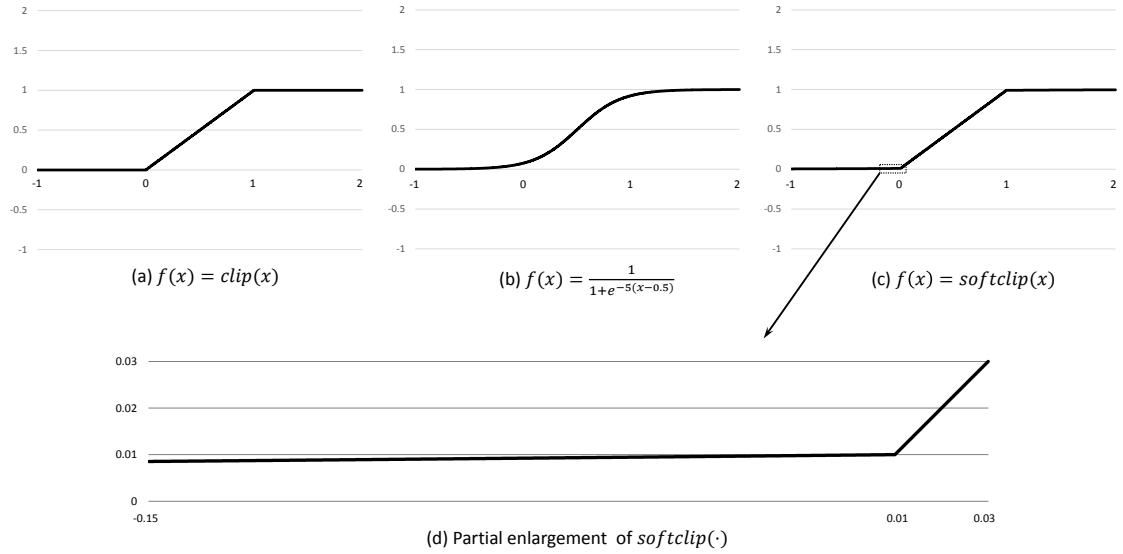

Figure 5: Comparison among clip function, logistic function, and softclip function

$softclip : \mathbb{R} \to (0, 1)$ is a non-decreasing, continuous function. However, It is not differentiable when $x = \epsilon$ or $x = 1 - \epsilon$. So we defined its derivative as follows:

$$\frac{\partial softclip(x)}{\partial x} \triangleq \begin{cases} \frac{\partial sigmoid(x-0.5+\delta)}{\partial x} & \text{if } x \leq \epsilon \\ 1 & \text{if } \epsilon < x < 1 - \epsilon \\ \frac{\partial sigmoid(x-0.5-\delta)}{\partial x} & \text{if } x \geq 1 - \epsilon \end{cases} \tag{20}$$

It is like a clip function. However, its derivative may be small on some inputs but is not zero. Figure 5 shows the comparison among clip function, logistic function, and softclip function. According to the result of an experiment done on the DSTC-2 dataset where RPNs using different activation functions are trained `dstc2trn` and tested on `dstc2dev`, softclip function has demonstrated better performance than both clip and logistic function[4], and is thus used in the rest of the paper.

With the activation function, a new type of computation node, referred to as *activation node*, is introduced. Activation node only takes one input and only has one input edge of type-2, i.e $|\hat{I}_x| = 1$ and $I_x = \emptyset$. The value of an activation node x is calculated as

$$u_x^{(t)} = softclip\left(u_{\jmath_x}^{(t)}\right) \tag{21}$$

where $\jmath_x$ denotes the input node of node x. i.e. $\hat{I}_x = \{\jmath_x\}$.

The activation function is used in the rest of the paper. Figure 6 gives an example of RPN with activation function, whose structure is constructed by adding an activation function to the RPN in figure 4.

4. The accuracy and L2 of RPNs with $clip$, $logistic$, and $softclip$ function are (0.779, 0.329), (0.789, 0.352), (0.790, 0.317) respectively. In particular, logistic functions with several different η are evaluated and the result reported here is the best one.

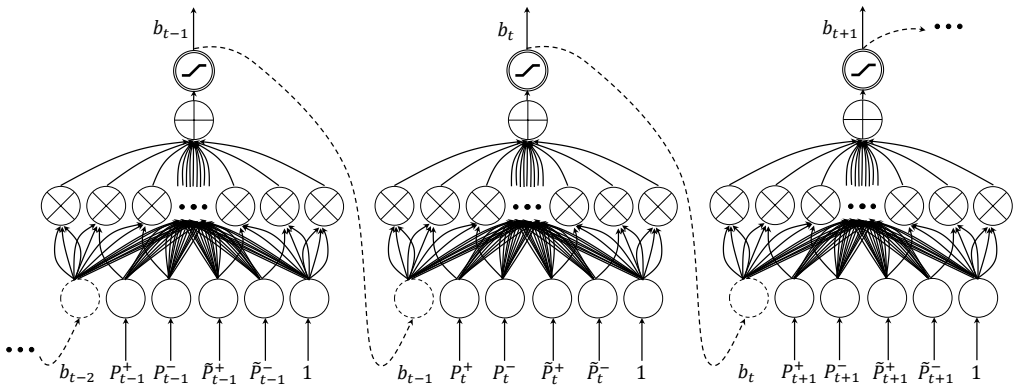

Figure 6: RPN for DST with activation functions

4.3 Further Exploration on Structure

Adding features to CMBP is not easy because additional prior knowledge is needed to add to keep the search space not too large. Concretely, adding features can introduce new monomials. Since the trivial search space is exponentially increasing as the number of monomials, the search space tends to be too large to explore when new features are added. Hence, to reduce the search space, additional prior knowledge is needed, which can introduce new constraints to the polynomial coefficients. For the same reason, increasing the model complexity also requires additional suitable prior knowledge to be added to limit the search space not too large in CMBP.

In contrast to that, since RPN can be seen as a data-driven model, it is as easy as most data-driven statistical approaches such as RNN to add new features to RPN and use more complex structures. At the same time, no matter what new features are used and how complex the structure is, RPN can always take advantage prior knowledge and intuition which is discussed in section 4.4. In this paper, both new features and complex structure are explored.

Adding new features can be done by just adding input nodes which correspond to the new features, and then adding product nodes corresponding to the new possible monomials introduced by the new features. In this paper, for slot s, value v at turn t, in addition to $f_0 \sim f_5$ which are defined as $b_{t-1}(v)$, $P^+_t(v)$, $P^-_t(v)$, $\tilde{P}^+_t(v)$, $\tilde{P}^-_t(v)$, and 1 respectively, 4 new features are investigated. f_6 and f_7 are features of system acts at the last turn: for slot s, value v at turn t,

- $f_6 \triangleq canthelp(s,t,v) \cup canthelp.missing\_slot\_value(s,t)$ =1 if the system cannot offer a venue with the constraint $s = v$ or the value of slot s is not known for the selected venue, otherwise 0.
- $f_7 \triangleq select(s,t,v)$ =1 if the system asks the user to pick a suggested value for slot s, otherwise 0.

f_6 and f_7 are introduced because user is likely to change their goal if given machine acts $canthelp(s,t,v)$, $canthelp.missing\_slot\_value(s,t)$ and $select(s,t,v)$. f_8 and f_9 are features of user acts at the current turn: for slot s, value v at turn t,

- $f_8 \triangleq inform(s,t,v)$ = 1 if one of SLU hypotheses from the user is informing slot s is v, otherwise 0.

- $f_9 \triangleq deny(s,t,v)$ =1 if one of SLU hypotheses from the user is denying slot s is v, otherwise 0.

f_8 and f_9 are features about SLU act type, introduced to make system robust when the confidence scores of SLU hypothesis are not reliable.

In this paper, the complexity of evaluating and training RPN for DST would not increase sharply because a constant order 3 is used and number of product nodes in the second layer grows from 56 to 220 when number of features grows from 6 to 10.

In addition to new features, RPN of more complex structure is also investigated in this paper. To capture some property just like belief b_t of dialogue process, a new sum node $x = (3,1)$ in the third layer is introduced. The connection of $(3,1)$ is the same as $(3,0)$, so it introduces a new recurrent connection. The exact meaning of its value is unknown. However, it is the only value used to record information other than b_t of previous turns. Every other input features except b_t are features of current turn t. Compared with b_t, there are fewer restrictions on the value of $(3,1)$ since its value is not directly supervised by the label. Hence, introducing $(3,1)$ may help to reduce the effect of inaccurate labels.

The structure of the RPN with 4 new features and 1 new sum node, together with new activation nodes introduced in section 4.2 is shown in figure 7.

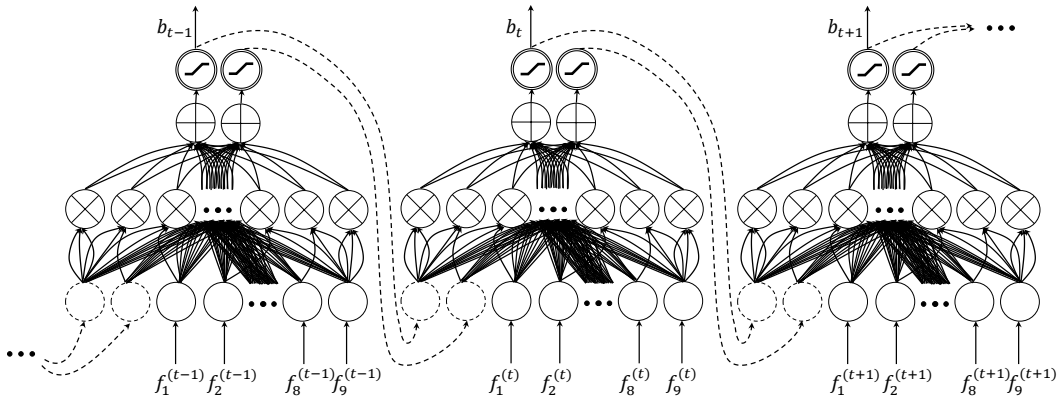

Figure 7: RPN with new features and more complex structure for DST

4.4 RPN Initialization

Like most neural network models such as RNN, the initialization of RPN can be done by setting each weight, i.e. w and $\hat{w}$, to be a small random value. However, with its unique structure, the initialization can be much better by taking advantage of the relationship between CMBP and RPN which is introduced in section 4.1.

When RPN is initialized according to a CMBP, prior knowledge and constraints are used to set RPN's initial parameters as a suboptimum point in the whole parameter space. RPN as a data-driven model can fully utilize the advantages of data-driven approaches. RPN is better than real CMBP while they both use data samples to train parameters. In the work of Yu et al. (2015), real-coefficient CMBP uses hill climbing to adjust parameters that are initially not zero and the change of parameters are always a multiple of 0.1. RPN can adjust all parameters including parameters initialized as 0 concurrently, while the complexity of adjusting all parameters concurrently is nearly

the same as adjusting one parameter in CMBP. Besides, the change of parameters can be large or small, depending on learning rate. Thus, RPN and CMBP both are combining rule-based models and data-driven ones, while RPN is a data-driven model utilizing rule advantages and CMBP is a rule model utilizing data-driven advantages.

In fact, given a CMBP, an RPN can achieve the same performance as the CMBP just by setting its weights according to the coefficients of the CMBP. To illustrate that, the steps of initializing the RPN in figure 7 with a CMBP of features $f_0 \sim f_9$ is described below.

First, to ensure that the new added sum node $x = (3, 1)$ will not influence the output b_t in RPN with initial parameters, $\hat{w}_{x,y}$ is set to 0 for all y. So node x's value $u_x^{(t)}$ is always 0.

Next, considering the RPN in figure 7 has more features than CMBP does, the weights related the new features should be set to 0. Specifically, suppose node x is the sum node in the third layer in RPN denoting b_t before activation and node y is one of the product nodes in the second layer denoting a monomial, if product node y is products of features f_6, f_7, f_8, f_9 or the added sum node, then node y's value is not a monomial in CMBP, then weights $\hat{w}_{x,y}$ should be set to 0.

Finally, if product node y is the product of features $f_0 \sim f_5$, suppose the order of CMBP is 3, then $\mathcal{F}(y) = (k_1, k_2, k_3)$ defined in equation (13) should satisfy $0 \leq k_1 \leq k_2 \leq k_3 \leq 5$. Weights $\hat{w}_{xy}$ should be initialized as g_{k_1,k_2,k_3} which is the coefficient of $f_{k_1} f_{k_2} f_{k_3}$ in CMBP. Thus,

$$w_{x,y} = \begin{cases} g_{k_1,k_2,k_3} & \text{if } x = (2, 0) \text{ and } \mathcal{F}(x) = (k_1, k_2, k_3) \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

For RPN of other structures, the initialization can be done by following similar steps.

Experiments show that after training, there are only a few weights larger than 0.1, no matter using CMBP or random initialization.

4.5 Training RPN

Suppose $T(d)$ is the number of turns in dialogue d, $H(d, s, t)$ is the set of values corresponding to slot s appearing in SLU hypothesis in turn t in dialogue d, $b_{d,s,t}(v)$ is the output belief of value v in dialogue d and $l_{d,s,t}(v)$ is the indicator of goal $s = v$ being part of joint goal at turn t in the label of dialogue d. The cost function is defined as

$$\mathcal{L} = \frac{1}{\sum_d T(d)} \sum_d \sum_{t=0}^{T(d)-1} \sum_s \sum_{v \in \cup_{i=0}^{t} H(d,s,i)} \left(b_{d,s,t}(v) - l_{d,s,t}(v)\right)^2 \tag{23}$$

Training process of a mini-batch can be divided into two parts: forward pass and backward pass.

Forward Pass For each training sample, every node's value at every time is evaluated first. When evaluating $u_x^{(t)}$, values of nodes in I_x and $\hat{I}_x$ should be evaluated before. The computation formula should be based on the type of node x. In particular, for a layered RPN structure, we can simply evaluate $u_{x_1}^{t_1}$ earlier than $u_{x_2}^{t_2}$ if $t_1 < t_2$ or $t_1 = t_2$ and x_1's layer number is smaller than x_2's.

Backward Pass Backpropagation through time (BPTT) is used in training RPN. Let error of node x at time t be $\delta_x^{(t)} = \frac{\partial \mathcal{L}}{\partial u_x^{(t)}}$. If a node x is an output node, then $\delta_x^{(t)}$ should be initialized according to its label l_t and output value $u_x^{(t)}$, otherwise $\delta_x^{(t)}$ should be initialized to 0. After a node's error

foreach *Mini batch* m **do**
  Initialize $\Delta w_{xy} = 0, \Delta \hat{w}_{x,y} = 0$ for every x, y
  Initialize the value of recurrent node at turn 0 as 0
  foreach *Training dialogue* d*, slot* s*, value* v *in mini batch* m **do**
    $T \leftarrow$ the number of turns of current training sample
    /* forward pass */
    for $t \leftarrow 1$ **to** T **do**
      for $d \leftarrow 1$ **to** 4 **do**
        foreach *node* x *in time* t*, layer* d **do**
          evaluate $u_x^{(t)}$
          if x *is output node* **then**
            $\delta_x^{(t)} \leftarrow 2(u_x^{(t)} - l_t)$
          else
            $\delta_x^{(t)} \leftarrow 0$
    /* backward pass */
    for $t \leftarrow T$ **to** 1 **do**
      for $d \leftarrow 4$ **to** 1 **do**
        foreach *node* x *in time* t*, layer* d **do**
          foreach *node* $y \in \hat{I}_x$ **do**
            /* Node y is linked to node x by a "type-2" edge */
            $\delta_y^{(t)} \leftarrow \delta_y^{(t)} + \delta_x^{(t)} \frac{\partial u_x^{(t)}}{\partial u_y^{(t)}}$
          foreach *node* $y \in I_x$ **do**
            /* Node y is linked to node x by a "type-1" edge */
            $\delta_y^{(t-1)} \leftarrow \delta_y^{(t-1)} + \delta_x^{(t)} \frac{\partial u_x^{(t)}}{\partial u_y^{(t-1)}}$
    for $t \leftarrow 1$ **to** T **do**
      for $d \leftarrow 1$ **to** 4 **do**
        foreach *sum node* x *in time* t*, layer* d **do**
          foreach *node* $y \in \hat{I}_x$ **do**
            $\Delta \hat{w}_{xy} \leftarrow \Delta \hat{w}_{xy} + \alpha \delta_x^{(t)} u_y^{(t)}$
          foreach *node* $y \in I_x$ **do**
            $\Delta w_{xy} \leftarrow \Delta w_{xy} + \alpha \delta_x^{(t)} u_y^{(t-1)}$
  foreach *edge*(x, y) **do**
    $w_{xy} \leftarrow w_{xy} - \Delta w_{xy}$
    $\hat{w}_{xy} \leftarrow \hat{w}_{xy} - \Delta \hat{w}_{xy}$

Algorithm 1: Training Algorithm of RPN for DST

$\delta_x^{(t)}$ is determined, it can be passed to $\delta_y^{(t-1)}(y \in I_x)$ and $\delta_y^{(t)}(y \in \hat{I}_x)$. Error passing should follow the reversed edge's direction. So the order of nodes passing error can follow the reverse order of evaluating nodes' values.

When every $\delta_x^{(t)}$ has been evaluated, the increment on weight $\hat{w}_{xy}$ can be calculated by

$$\begin{aligned}\Delta\hat{w}_{xy} &= \alpha\frac{\partial\mathcal{L}}{\partial\hat{w}_{xy}} \\ &= \alpha\sum_{i=1}^{T}\frac{\partial\mathcal{L}}{\partial u_x^{(t)}}\frac{\partial u_x^{(t)}}{\partial\hat{w}_{xy}} \\ &= \alpha\sum_{i=1}^{T}\delta_x^{(t)}u_y^{(t)}\end{aligned} \tag{24}$$

where α is the learning rate. Δw_{xy} can be evaluated similarly.

Note that only w_{xy} and $\hat{w}_{xy}$ are parameters of RPN.

The complete formula of evaluating node value $u_x^{(t)}$ and passing error $\delta_x^{(t)}$ can be found in appendix.

In this paper, mini-batch is used in training RPN for DST with batch size 8. In each training epoch, Δw_{xy} and $\Delta\hat{w}_{xy}$ are calculated for every training sample and added together. The weight w_{xy} and $\hat{w}_{xy}$ is updated by

$$w_{xy} = w_{xy} - \Delta w_{xy} \tag{25}$$

$$\hat{w}_{xy} = \hat{w}_{xy} - \Delta\hat{w}_{xy} \tag{26}$$

The pseudocode of training is shown in algorithm 1.

5. Experiment

As introduced in section 1, in this paper, DSTC-2 and DSTC-3 tasks are used to evaluate the proposed approach. Both tasks provide training dialogues with turn-level ASR hypotheses, SLU hypotheses and user goal labels. The DSTC-2 task provides 2118 training dialogues in restaurants domain (Henderson et al., 2014b), while in DSTC-3, only 10 in-domain training dialogues in tourists domain are provided, because the DSTC-3 task is to adapt the tracker trained on DSTC-2 data to the new domain with very few dialogues (Henderson et al., 2014a). Table 1 summarizes the size of datasets of DSTC-2 and DSTC-3.

Task	Dataset	#Dialogues	Usage
DSTC-2	`dstc2trn`	1612	Training
	`dstc2dev`	506	Training
	`dstc2eval`	1117	Test
DSTC-3	`dstc3seed`	10	Not used
	`dstc3eval`	2265	Test

Table 1: Summary of data corpora of DSTC-2/3

The DST evaluation criteria are the joint goal *accuracy* and the *L2* (Henderson et al., 2014b,a). *Accuracy* is defined as the fraction of turns in which the tracker's 1-best joint goal hypothesis is

correct, the larger the better. *L2* is the L2 norm between the distribution of all hypotheses output by the tracker and the correct goal distribution (a delta function), the smaller the better. Besides, schedule 2 and labelling scheme A defined in (Henderson et al., 2013) are used in both tasks. Specifically, schedule 2 only counts the turns where new information about some slots either in a system confirmation action or in the SLU list is observed. Labelling scheme A is that the labelled state is accumulated forwards through the whole dialogue. For example, the goal for slot s is "`None`" until it is informed as $s = v$ by the user, from then on, it is labelled as v until it is again informed otherwise.

It has been shown that the organiser-provided live SLU confidence was, at best, a noisy signal (Zhu et al., 2014; Sun et al., 2014a). Hence, most of the state-of-the-art results from DSTC-2 and DSTC-3 used refined SLU (either explicitly rebuild a SLU component or take the ASR hypotheses into the trackers (Williams, 2014; Sun et al., 2014a; Henderson et al., 2014d,c; Kadlec et al., 2014; Sun et al., 2014b)). In accordance to this, except for the results directly taken from other papers (shown in table 5 and 6), all experiments in this paper used the output from a refined semantic parser (Zhu et al., 2014; Sun et al., 2014a) instead of the live SLU provided by the organizer.

For all experiments, MSE is used as the training criterion. MSE is chosen because of two reasons: (i) MSE can directly reflect L2 performance which is one of the main metrics in DSTCs. (ii) Experiment has shown that some other criterion such as cross-entropy loss cannot lead to better performance. For both DSTC-2 and DSTC-3 tasks, `dstc2trn` and `dstc2dev` are used, 60% of the data is used for training and 40% for validation, unless otherwise stated. Validation is performed every 5 epochs. Learning rate is set to 0.6 initially. During the training, learning rate is halved each time the performance does not increases. Training is stopped when the learning rate is sufficiently small, or the maximum number of training epochs is reached. Here, the maximum number of training epochs is set to 40. L2 regularization is used for all the experiments. The parameter of L2 regularization is set to be the one leading to the best performance on `dstc2dev` when trained on `dstc2trn`. L1 regularization is not used since it cannot yield better performance.

5.1 Investigation on RPN Configurations

This section describes the experiments comparing different configurations of RPN. All experiments were performed on both the DSTC-2 and DSTC-3 tasks.

As indicated in section 4.4, an RPN can be initialized by a CMBP. Table 2 shows the performance comparison between initialization with a CMBP and with random values. In this experiment, the structure shown in figure 6 is used. The performance of initialization with random values reported here is the average performance of 10 different random seeds, and their standard deviations are given in parentheses. The random scheme used is the one with the best performance on `dstc2dev` when trained on `dstc2trn` among various kinds of random initialization schemes.

The performance of the RPN initialized by random values is compared with the performance of the RPN initialized by the integer-coefficient CMBP. Here, the CMBP has 12 non-zero coefficients and has the best performance on `dstc2dev` when trained on `dstc2trn`. It can be seen from table 2 that the RPN initialized by the CMBP coefficients outperforms the RPN initialized by random values moderately on `dstc2eval` and significantly on `dstc3eval` (p-value< 0.05). This demonstrates the encoded prior knowledge and intuition in CMBP can be transferred to RPN to improve RPN's performance, which is one of RPN's advantage, combining rule-based models and

Initialization	dstc2eval		dstc3eval	
	Acc	L2	Acc	L2
Random	0.753 (0.008)	0.468 (0.020)	0.633 (0.005)	0.667 (0.026)
CMBP	0.756	0.373	0.648	0.553

Table 2: Performance comparison between the RPN initialized by random values, and the RPN initialized by the CMBP coefficients on dstc2eval and dstc3eval.

data-driven models. In the rest of the experiments, all RPNs use CMBP coefficients for initialization.

Since section 4.3 shows that it is convenient to add features and try more complex structures, it is interesting to investigate RPNs with different feature sets and structures, as shown in table 3. It can be seen that while no obvious correlation between the performance and different configurations of feature sets and structures can be observed on dstc2eval and dstc3eval, RPNs with new features and new recurrent connections have achieved slightly better performance on accuracy, though the performance difference is not significant. In the rest of the paper, both new features and new recurrent connections are used in RPN, unless otherwise stated.

Feature Set	New Recurrent Connections	dstc2eval		dstc3eval	
		Acc	L2	Acc	L2
$f_0 \sim f_5$	No	0.756	0.373	0.648	0.553
$f_0 \sim f_9$		0.757	0.374	0.650	0.557
$f_0 \sim f_5$	Yes	0.756	0.373	0.648	0.553
$f_0 \sim f_9$		0.757	0.374	0.650	0.549

Table 3: Performance comparison among RPNs with different configurations on dstc2eval and dstc3eval.

5.2 Comparison with Other DST Approaches

The previous subsection investigates how to get the RPN with the best configuration. In this subsection, the performance of RPN is compared to both rule-based and data-driven statistical approaches. To make fair comparison, all data-driven models together with RPN in this subsection use similar feature set. Altogether, 2 rule-based trackers and 3 data-driven trackers were built for performance comparison.

- **MaxConf** is a *rule-based* model commonly used in spoken dialogue systems which always selects the value with the highest confidence score from the 1^{st} turn to the current turn. It was used as one of the primary baselines in DSTC-2 and DSTC-3.

- **HWU** is a *rule-based* model proposed by Wang and Lemon (2013). It is regarded as a simple, yet competitive baseline of DSTC-2 and DSTC-3.

Type	System	dstc2eval		dstc3eval	
		Acc	L2	Acc	L2
Rule	MaxConf	0.668	0.647	0.548	0.861
	HWU	0.720	0.445	0.594	0.570
Data-driven	DNN	0.719	0.469	0.628	0.556
	MaxEnt	0.710	0.431	0.607	0.563
	LSTM	0.736	0.418	0.632	0.549
Hybrid	CMBP	0.755	0.372	0.627	0.546
	RPN	0.756	0.373	0.648	0.553

Table 4: Performance comparison among RPN, rule-based and data-driven approaches with similar feature set on `dstc2eval` and `dstc3eval`. The performance of CMBP in the table is the performance of the RPN which has been initialized but not been trained.

- **DNN** is a *data-driven statistical model* using deep neural network model (Sun et al., 2014a) with probability feature as RPN. Since DNN does not have recurrent structures while RPN does, to fairly take into account this, the DNN feature set at the t^{th} turn is defined as

$$\bigcup_{i\in\{t-9,\cdots,t\}} \left\{P_i^+(v), P_i^-(v), \tilde{P}_i^+(v), \tilde{P}_i^-(v)\right\} \cup \left\{\hat{P}(t)\right\}$$

  where $\hat{P}(t)$ is the highest confidence score from the 1^{st} turn to the t^{th} turn. The DNN has 3 hidden layers with 64 nodes per layer.

- **MaxEnt** is also a *data-driven statistical model* using Maximum Entropy model (Sun et al., 2014a) with the same input features as DNN.

- **LSTM** is another *data-driven statistical model* using long short-term memory model (Hochreiter and Schmidhuber, 1997) with the same input features as RPN. It has similar structure as the DNN model (Sun et al., 2014a) except for its hidden layers using LSTM blocks. The LSTM used here has 3 hidden layers with 100 LSTM blocks per layer.

It can be observed that, with similar feature set, RPN can outperform both rule-based and data-driven approaches in terms of joint goal accuracy. Statistical significance tests were also performed assuming a binomial distribution for each turn. RPN was shown to significantly outperform both rule-based and data-driven statistical approaches at 95% confidence level. For L2, RPN is competitive to both rule-based and the data-driven statistical approaches.

5.3 Comparison with State-of-the-art DSTC Trackers

In the DSTCs, the state-of-the-art trackers mostly employed data-driven statistical approaches. Usually, richer feature set and more complicated model structures than the data-driven statistical models in section 5.2 are used. In this section, the proposed RPN approach is compared to the best submitted trackers in DSTC-2/3 and the best CMBP trackers, regardless of fairness of feature selection and the SLU refinement approach. RPN is compared and the results are shown in table 5 and table

System	Approach	Rank	Acc	L2
Baseline*	Rule	5	0.719	0.464
Williams (2014)	LambdaMART	1	0.784	0.735
Henderson et al. (2014d)	RNN	2	0.768	0.346
Sun et al. (2014a)	DNN	3	0.750	0.416
Yu et al. (2015)	Real CMBP	2.5	0.762	0.436
RPN	RPN	2.5	0.757	0.374

Table 5: Performance comparison among RPN, real-coefficient CMBP and best trackers of DSTC-2 on `dstc2eval`. Baseline* is the best results from the 4 baselines in DSTC2.

6. Note that structure shown in figure 7 with richer feature set and a new recurrent connection is used here.

Note that, in DSTC-2, the Williams (2014)'s system employed batch ASR hypothesis information (i.e. off-line ASR re-decoded results) and cannot be used in the normal on-line model in practice. Hence, the practically best tracker is Henderson et al. (2014d). It can be observed from table 5, RPN ranks only second to the best practical tracker among the submitted trackers in DSTC-2 in accuracy and L2. Considering that RPN only used probabilistic features and very limited added features and can operate very efficiently, it is quite competitive.

System	Approach	Rank	Acc	L2
Baseline*	Rule	6	0.575	0.691
Henderson et al. (2014c)	RNN	1	0.646	0.538
Kadlec et al. (2014)	Rule	2	0.630	0.627
Sun et al. (2014b)	Int CMBP	3	0.610	0.556
Yu et al. (2015)	Real CMBP	1.5	0.634	0.579
RPN	RPN	0.5	0.650	0.549

Table 6: Performance comparison among RPN, real-coefficient CMBP and best trackers of DSTC-3 on `dstc3eval`. Baseline* is the best results from the 4 baselines in DSTC3.

It can be seen from table 6, RPN trained on DSTC-2 can achieve state-of-the-art performance on DSTC-3 without modifying tracking method[5], outperforming all the submitted trackers in DSTC-3 including the RNN system. This demonstrates that RPN successfully inherits the advantage of good generalization ability of rule-based model. Considering the feature set and structure of RPN are relatively simple in this paper, future work will investigate richer features and more complex structures.

6. Conclusion

This paper proposes a novel hybrid framework, referred to as recurrent polynomial network, to improve the combination of the rule-based approaches and data-driven approaches. With the ability

5. The parser is refined for DSTC-3 (Zhu et al., 2014).

of incorporating prior knowledge into a data-driven framework, RPN has the advantages of both rule-based and data-driven approaches. Experiments on two DSTC tasks showed that the proposed approach not only is more stable than many major data-driven statistical approaches, but also has competitive performance, outperforming many state-of-the-art trackers.

Since the RPN in this paper only used probabilistic features and very limited added features, the performance of RPN can be influenced by how reliable the SLU's confidence scores are. Therefore, future work will investigate the influence of SLUs on the performance of RPN, and rich features for RPN. Moreover, future work will also address applying RPN to other domains, such as the bus timetables domain in DSTC-1, and theoretic analysis of RPN.

References

Ian Cloete and Jacek M Zurada. *Knowledge-based neurocomputing*. MIT press, 2000.

Li Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Michael Seltzer, Geoff Zweig, Xiaodong He, Jason Williams, et al. Recent advances in deep learning for speech research at Microsoft. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8604–8608. IEEE, 2013.

Milica Gašić and Steve Young. Effective handling of dialogue state in the hidden information state POMDP-based dialogue manager. *ACM Transactions on Speech and Language Processing (TSLP)*, 7(3):4, 2011.

Matthew Henderson, Blaise Thomson, and Jason Williams. Dialog state tracking challenge 2 & 3. 2013.

Matthew Henderson, Blaise Thomson, and Jason D. Williams. The third dialog state tracking challenge. In *Proceedings of IEEE Spoken Language Technology Workshop (SLT)*, December 2014a.

Matthew Henderson, Blaise Thomson, and Jason D Williams. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272, Philadelphia, PA, U.S.A., June 2014b. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W14-4337`.

Matthew Henderson, Blaise Thomson, and Steve Young. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *Proceedings of IEEE Spoken Language Technology Workshop (SLT)*, December 2014c.

Matthew Henderson, Blaise Thomson, and Steve Young. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 292–299, Philadelphia, PA, U.S.A., June 2014d. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W14-4340`.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Rudolf Kadlec, Miroslav Vodoln, Jindrich Libovick, Jan Macek, and Jan Kleindienst. Knowledge-based dialog state tracking. In *Proceedings 2014 IEEE Spoken Language Technology Workshop*, South Lake Tahoe, USA, December 2014.

Sungjin Lee. Structured discriminative model for dialog state tracking. In *Proceedings of the SIGDIAL 2013 Conference*, pages 442–451, Metz, France, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W13/W13-4069.

Sungjin Lee and Maxine Eskenazi. Recipe for building robust spoken dialog state trackers: Dialog state tracking challenge system description. In *Proceedings of the SIGDIAL 2013 Conference*, pages 414–422, Metz, France, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W13/W13-4066.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Nicholas Roy, Joelle Pineau, and Sebastian Thrun. Spoken dialogue management using probabilistic reasoning. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 93–100. Association for Computational Linguistics, 2000.

Kai Sun, Lu Chen, Su Zhu, and Kai Yu. The SJTU system for dialog state tracking challenge 2. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 318–326, Philadelphia, PA, U.S.A., June 2014a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W14-4343.

Kai Sun, Lu Chen, Su Zhu, and Kai Yu. A generalized rule based tracker for dialogue state tracking. In *Proceedings of IEEE Spoken Language Technology Workshop (SLT)*, December 2014b.

Blaise Thomson and Steve Young. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562–588, 2010.

Zhuoran Wang and Oliver Lemon. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *Proceedings of the SIGDIAL 2013 Conference*, pages 423–432, Metz, France, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W13/W13-4067.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413, Metz, France, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W13/W13-4065.

Jason D Williams. Challenges and opportunities for state tracking in statistical spoken dialog systems: Results from two public deployments. *Selected Topics in Signal Processing, IEEE Journal of*, 6(8):959–970, 2012.

Jason D Williams. Web-style ranking and SLU combination for dialog state tracking. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 282–291, Philadelphia, PA, U.S.A., June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W14-4339.

Jason D Williams and Steve Young. Scaling up pomdps for dialog management: The"summary pomdp"method. In *Automatic Speech Recognition and Understanding, 2005 IEEE Workshop on*, pages 177–182. IEEE, 2005.

Jason D Williams and Steve Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422, 2007.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. The hidden information state model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174, 2010.

Kai Yu, Kai Sun, Lu Chen, and Su Zhu. Constrained markov bayesian polynomial for efficient dialogue state tracking. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(12):2177–2188, December 2015.

Bo Zhang, Qingsheng Cai, Jianfeng Mao, Eric Chang, and Baining Guo. Spoken dialogue management as planning and acting under uncertainty. In *INTERSPEECH*, pages 2169–2172, 2001.

Su Zhu, Lu Chen, Kai Sun, Da Zheng, and Kai Yu. Semantic parser enhancement for dialogue domain extension with little data. In *Proceedings of IEEE Spoken Language Technology Workshop (SLT)*, December 2014.

Lukas Zilka, David Marek, Matej Korvas, and Filip Jurcicek. Comparison of bayesian discriminative and generative models for dialogue state tracking. In *Proceedings of the SIGDIAL 2013 Conference*, pages 452–456, Metz, France, August 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W13/W13-4070`.

Appendix

Derivative calculation

Using MSE as the criterion, $\delta_x^{(t)} = \frac{\partial \mathcal{L}}{\partial u_x^{(t)}}$ is initialized as following[6]:

$$\delta_x^{(t)} = \begin{cases} 2(u_x^{(t)} - l_{d,s,t}(v)) & \text{if } x \text{ is a output node calculating } b_{d,s,t}(v) \\ 0 & \text{otherwise} \end{cases} \tag{27}$$

Suppose node x is an activation node and $f(\cdot) = softclip(\cdot)$, let $y = \jmath_x$,

$$\begin{aligned} \delta_y^{(t)} &= \frac{\partial \mathcal{L}}{\partial u_y^{(t)}} \\ &= \frac{\partial \mathcal{L}}{\partial u_x^{(t)}} \frac{\partial u_x^{(t)}}{\partial u_y^{(t)}} \\ &= \delta_x^{(t)} \frac{\partial f(u_y^{(t)})}{\partial u_y^{(t)}} \end{aligned} \tag{28}$$

6. The symbols used in this section such as $\hat{I}$, I, $\hat{M}$, M follow the definitions in section 3.

Suppose node $x = (d, i)$ is a sum node, then when node x passes its error, the error of node $y \in \hat{I}_x$ is updated as

$$\begin{aligned} \delta_y^{(t)} &= \delta_y^{(t)} + \frac{\partial \mathcal{L}}{\partial u_x^{(t)}} \frac{\partial u_x^{(t)}}{\partial u_y^{(t)}} \\ &= \delta_y^{(t)} + \delta_x^{(t)} \hat{w}_{x,y} \end{aligned} \tag{29}$$

Similarly, error of node $y \in I_x$ is updated as

$$\begin{aligned} \delta_y^{(t)} &= \delta_y^{(t)} + \frac{\partial \mathcal{L}}{\partial u_x^{(t)}} \frac{\partial u_x^{(t)}}{\partial u_y^{(t-1)}} \\ &= \delta_y^{(t)} + \delta_x^{(t)} w_{x,y} \end{aligned} \tag{30}$$

Suppose node $x = (d, i)$ is a product node, then when node x passes its error, error of node $y \in \hat{I}_x$ is updated as

$$\begin{aligned} \delta_y^{(t)} &= \delta_y^{(t)} + \frac{\partial \mathcal{L}}{\partial u_x^{(t)}} \frac{\partial u_x^{(t)}}{\partial u_y^{(t)}} \\ &= \delta_y^{(t)} + \delta_x^{(t)} \hat{M}_{x,y} {u_y^{(t)}}^{\hat{M}_{x,y}-1} \prod_{z \in \hat{I}_x - \{y\}} {u_z^{(t)}}^{\hat{M}_{x,z}} \prod_{z \in I_x} {u_z^{(t-1)}}^{M_{x,z}} \end{aligned} \tag{31}$$

Similarly, error of node $y \in I_x$ is updated as

$$\begin{aligned} \delta_y^{(t)} &= \delta_y^{(t)} + \frac{\partial \mathcal{L}}{\partial u_x^{(t)}} \frac{\partial u_x^{(t)}}{\partial u_y^{(t-1)}} \\ &= \delta_y^{(t)} + \delta_x^{(t)} M_{x,y} {u_y^{(t-1)}}^{M_{x,y}-1} \prod_{z \in \hat{I}_x} {u_z^{(t)}}^{\hat{M}_{x,z}} \prod_{z \in I_x - \{y\}} {u_z^{(t-1)}}^{M_{x,z}} \end{aligned} \tag{32}$$